

Spectral Analysis

FFTs and Beyond

Digital signal processors are often used for doing spectral analysis of incoming waveforms, but the old standby FFT can have its downside in the real world. Find out what to watch for in your next design.

FEATURE ARTICLE

David Prutchi

t

he analysis of a signal based on its frequency content is commonly referred to as *spectral analysis*. Although the mathematical basis for this operation, the Fourier Transform, has been known for many years, it was the introduction of the Fast Fourier Transform (FFT) algorithm which made spectral analysis a practical reality.

Implementing the FFT in personal computers and embedded DSP systems has offered an efficient and economical application of Fourier techniques to a wide variety of measurement and analysis tasks. Moreover, because the FFT has been found to be so valuable in applications such as medical signal

processing, radar, and telecommunications, DSP chips are often designed to implement the FFT with the greatest efficiency.

In most instances, the powerful Fourier techniques, used in modems, fax machines, and CT or ultrasound scanners, are hidden from the user, who doesn't have to worry about their mathematical implications. In other cases, however, human interpreters must make diagnostic decisions based on frequency-domain representations of data processed through Fourier transforms.

For example, many digital storage oscilloscopes offer the user the option of converting time-domain signals into the frequency domain through the use of the FFT which runs on an embedded DSP and displays results directly on screen. It is also common for scientists and engineers to write short FFT-based routines to display a spectral representation of experimental data acquired by a personal computer. It is in these cases where the unwary may fall into one of the many traps that the FFTs conceal.

FFT users often forget that real-world signals are seldom periodic, free of noise and distortion, and that signal and noise statistics play an important role in their analysis. Because of these

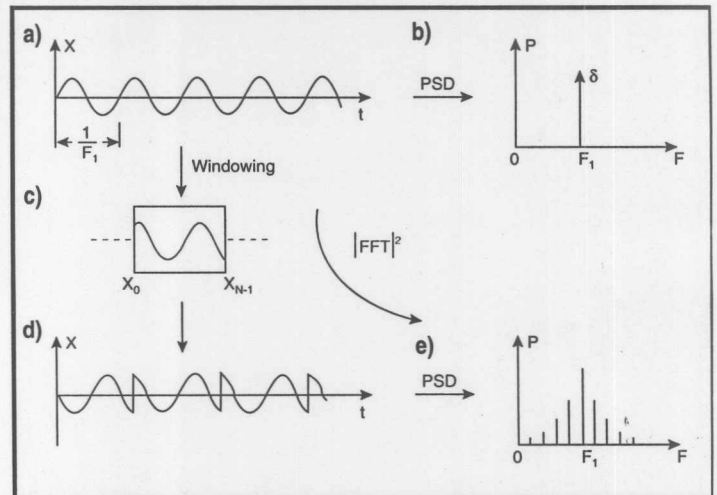


Figure 1—A purely sinusoidal signal (a) has a single impulse as its true spectrum (b). However, the signal is viewed by FFT through a finite window (c), and it is assumed that this record is repeated beyond the FFT's window (d). This leads to leakage of the main lobe to sidelobes in the spectral estimate (e).

"problem" factors, the FFTs and other methods can only provide estimates of the actual spectrum of signals. The results require competent interpretation by the user for correct analysis.

In this article, I will explain the common pitfalls in the use of the FFT and how to avoid them. After exposing some of the inherent problems which make the FFT unsuitable for high-resolution applications, I'll present more powerful spectral estimation methods, which cope with the fundamental shortcomings of the FFT, and describe typical applications for these methods.

FFTs AND THE POWER SPECTRAL DENSITY

Using a typical data acquisition setup, a signal is sampled at a fixed rate of

$$f_s = \left[\frac{\text{samples}}{\text{second}} \right]$$

which yields discrete data samples $x_0, x_1, \dots, x_{N-1}$. These N samples are then equally spaced by the discrete sampling period $\Delta t [\text{seconds}] = 1/f_s$. The discrete Fourier transform (DFT) represents the time-domain data with N -spaced samples in the frequency domain $X_0, X_1, \dots, X_{N-1}$ through:

$$X(f) = \Delta t \sum_{n=0}^{N-1} x_n e^{-2\pi j f n \Delta t} \quad (1)$$

where the frequency $f [\text{Hz}]$ is defined over the interval $-1/2\Delta t \leq f \leq 1/2\Delta t$. The FFT efficiently evaluates this expression at a discrete set of N frequencies spaced equally by $\Delta f [\text{Hz}] = 1/N\Delta t$.

In its most simple form, the energy-spectral-density estimate of the time-domain data is given by the squared modulus of this data's FFT, and the power spectral density (PSD) estimate $P(f)$ at every discrete frequency f is obtained by dividing the latter by the time interval $N\Delta t$:

$$P(f_m) = \frac{|X_m|^2}{N\Delta t}; m = 0, 1, \dots, N-1 \quad (2)$$

where $f_m [\text{Hz}] = m\Delta f$. In a case which uses real data (this is the norm when sampling from real-world signals), the PSD for negative frequencies is symmetrical to the PSD for positive frequencies, making only half of the

Window	3 dB Bandwidth [bins]	Scalloping Loss [dB]	Highest Sidelobe [dB]
Rectangular			
$w(n) = \begin{cases} 1 & \text{for } -\frac{N}{2} \leq n \leq \frac{N}{2} - 1 \\ 0 & \text{for all other} \end{cases}$	0.89	3.92	-13
Triangular			
$w(n) = \begin{cases} 1 - \frac{ 2n+1 }{N} & \text{for } -\frac{N}{2} \leq n \leq \frac{N}{2} - 1 \\ 0 & \text{for all other} \end{cases}$	1.28	1.82	-27
Hamming			
$w(n) = \begin{cases} 0.54 + 0.46 \cos\left(\frac{2\pi(2n+1)}{2(N-1)}\right) & \text{for } -\frac{N}{2} \leq n \leq \frac{N}{2} - 1 \\ 0 & \text{for all other} \end{cases}$	1.30	1.78	-43
Hanning			
$w(n) = \begin{cases} 0.5 + 0.5 \cos\left(\frac{2\pi(2n+1)}{2(N-1)}\right) & \text{for } -\frac{N}{2} \leq n \leq \frac{N}{2} - 1 \\ 0 & \text{for all other} \end{cases}$	1.44	1.42	-32

Table 1—Typical window functions for use with the FFT in spectral estimation. Windows are N points long and are assumed here to be symmetric around $n = 0$.

PSD useful. However at times, it may be necessary to compute PSD for complex data where relevant results are obtained for both positive and negative frequencies.

Although obtaining the PSD seems to be as simple as computing the FFT and obtaining the square modulus of the results, it must be noted that, because the data set employed to obtain the Fourier transform is a limited record of the actual data series, the PSD obtained is only an estimate of the true PSD. Moreover, as will be seen later, meaningless spectral estimates may be obtained by using Equation (2) without performing some kind of statistical averaging of the PSD.

PITFALLS OF THE FFT

When sampling a continuous signal, information may be lost because no data is available between the sample points. As the sampling rate is increased, a larger portion of the information is made available. According to Nyquist's theorem, to correctly sample a waveform, the sampling rate must be at least twice that of the

highest-frequency component of the waveform. Disregarding this rule will result in aliasing—a process in which signal components of frequency higher than half the sampling rate appear as components with a frequency equal to the difference between the actual frequency of the component and the sampling rate.

Because aliased components cannot be distinguished from real signals after sampling, aliasing is not just a minor source of error. It is therefore of extreme importance that antialiasing filters with very high roll-off be used for all serious spectral analysis.

Beyond appropriate sampling practices, the FFT still exposes other inherent traps which can potentially prevent analysis of a signal. The most important problems include leakage and the picket-fence effect.

Leakage is caused by the fact that the FFT works on a short portion of the signal, a phenomenon called *windowing*, because the FFT can only see the portion of the signal that falls within its sampling "window," after which it assumes that windowed data

repeats itself indefinitely. However, as shown in Figure 1, this assumption is only seldom correct. In most cases, the FFT analyzes a distorted version of the signal that contains discontinuities resulting from appending windowed data to their duplicates. In PSD, these discontinuities appear as a leakage of the energy's real frequency components into sidelobes which show up on either side of a peak.

The second problem, called the *picket-fence effect* or scalloping, is inherently related to the discrete nature of the DFT. That is, the DFT calculates the frequency content of a signal at very well-defined discrete points in the frequency domain rather than producing a continuous spectrum. In a perfect system, if a certain component of the signal had a frequency falling between the discrete frequencies computed by the DFT, this component would not appear in the estimated PSD.

To visualize this problem, suppose that an ideal signal is sampled at a rate of 2048 Hz and processed through a

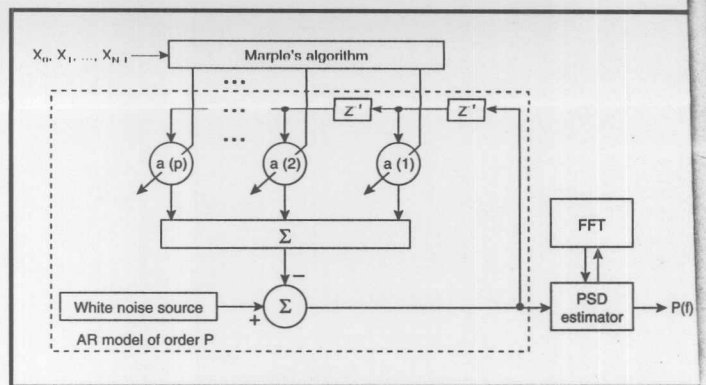


Figure 2—In one implementation of a parametric high-resolution spectral estimator, the coefficients $a_1, a_2, \dots, a_p$ of an AR filter are determined from input data through Marple's algorithm. The transfer function of the filter is then evaluated efficiently by FFT, resulting in a high-resolution estimate of the input data's PSD.

256-point FFT. There would be a spectral channel every 4 Hz (at DC, 4 Hz, 8 Hz, 12 Hz, etc.). Suppose now that the signal being analyzed is a pure sinusoidal with a frequency of 10 Hz. In a perfect system, this signal would not appear in the PSD because it falls between two discrete frequency channels—much like the case of a

picket fence which allows us to see detail in the scene behind it only if the details happen to fall within a slot between the boards.

In reality, however, because the FFT produces slightly overlapping "bins" of finite bandwidth, components with frequencies that fall between the theoretical discrete lines

BCC52 BASIC-52 Computer/Controller

The BCC52 controller continues to be Micromint's best selling single-board computer. Its cost-effective architecture needs only a power supply and terminal to become a complete development system or single-board solution in an end-use system. The BCC52 is programmable in BASIC-52, (a fast, full floating point interpreted BASIC), or assembly language.

The BCC52 contains five RAM/ROM sockets, an "intelligent" 2764/128 EPROM programmer, three 8-bit parallel ports, an auto-baud rate detect serial console port, a serial printer port, and much more.

PROCESSOR

- 80C52 8-bit CMOS processor w/BASIC-52
- Three 16-bit counter/timers
- Six interrupts
- Much more!

MEMORY

- 48K RAM/ROM, expandable
- Five on-board memory sockets
- Either 8K or 16K EPROM

Input/Output

- Console RS232 - autobaud detect
- Line printer RS-232
- Three 8-bit parallel ports
- EXPANDABLE!
- Compatible with 12 BCC expansion boards

To Order Call 1-800-635-3355

Tel: (203) 871-6170

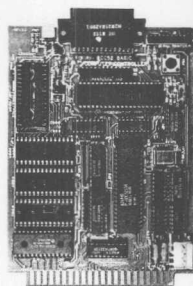
Fax: (203) 872-2204

BCC52	Controller board with BASIC-52 and 8K RAM	\$189.00 Single Qty.
BCC52C	Low-power CMOS version of the BCC52	\$199.00
BCC52I	-40°C to +85°C industrial temperature version	\$294.00
BCC52CX	Low-power CMOS, expanded BCC52 w/32K RAM	\$259.00

CALL FOR OEM PRICING



MICROMINT, INC. 4 Park Street, Vernon, CT 06066
in Europe: (44) 0285-658122 • in Canada: (514) 336-9426 • in Australia: (3) 467-7194
Distributor Inquiries Welcome!



CONCEPT TO MARKET

Professional Computer Services Specializing in System Design and Software

- ➡ 8051 and family
- ➡ PIC16C5X and PIC16C71/84
- ➡ 386 + /Windows 3.1
- ➡ C, C++, BASIC, ASM
- ➡ Real-Time Embedded Control
- ➡ Data Acquisition, Automation
- ➡ Communications, etc.

Satisfaction Guaranteed

MYRIAD DEVELOPMENT Co.

9220 West Tennessee Ave.

Lakewood, CO 80226

(303) 692-3836

are distributed among adjacent bins, but at reduced magnitudes. This attenuation is the actual picket-fence or scalloping error. Both of these problems are somewhat corrected by the use of an appropriate window.

So far, all samples presented to the FFT have been considered equal, which means that a weight of one has been implicitly applied to all samples. The samples outside of the FFT's scope are not considered, and thus their effective weight is zero, resulting in a rectangular-shaped window. This ultimately leads to the discontinuities that cause leakage.

A number of windows have been devised which reduce the amplitude of the samples at the edges of the window while increasing the relevance of samples towards its center. By doing so, these windows reduce the discontinuity to zero, thus lowering the amplitude of the sidelobes that surround a peak in the PSD. In addition, the use of a nonrectangular window increases the bandwidth of each bin, which results in a decreased scalloping error.

Some typical window functions and their characteristics are presented in Table 1. In essence, these functions produce N weights $w_0, w_1, \dots, w_{N-1}$ which are "weighted" (multiplied) one-to-one with their corresponding data samples $x_0, x_1, \dots, x_{N-1}$ before subjecting them to the FFT:

$$X(f) = \Delta t \sum_{n=0}^{N-1} w_n x_n e^{-2\pi j f n \Delta t} \quad (3)$$

Reduced resolution is the price paid for a reduction in leakage and scalloping through the use of a nonrectangular window. In fact, if it is necessary to view two closely spaced peaks, the rectangular window's narrow main lobe lets the user obtain analysis results, which report the existence of these closely spaced components. Any of the other windows would end up fusing these two peaks into a single smooth crest.

The use of a rectangular window is also appropriate for the analysis of transients. In these cases, a zero signal usually precedes and succeeds the transient. Thus, if the FFT is forced to look at the complete data record for

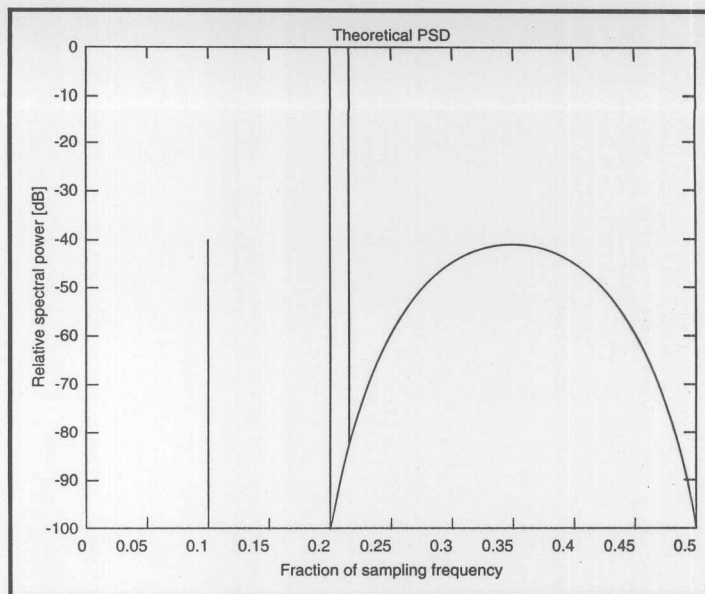


Figure 3a—Positive frequency theoretical spectrum of Marple's 64-point complex data test set. This spectrum includes features that are well suited for evaluating spectral estimators.

the transient, no artificial discontinuities are introduced, and full resolution can be obtained without leakage.

As you see, there is no single window which outperforms all others in every respect, and it is safe to say

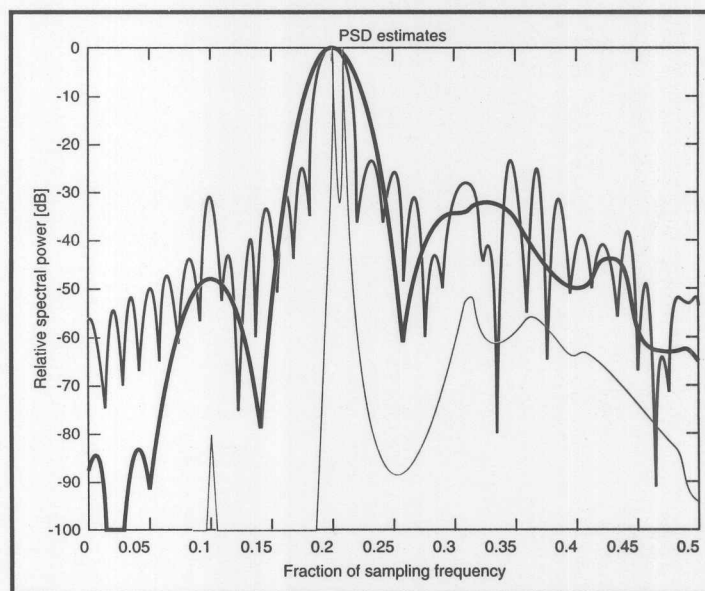


Figure 3b—Unlike the very well-established features of the theoretical spectrum shown in Figure 3a, spectral estimates distort the PSD according to their inherent assumptions. The spectrum is estimated using three different methods: i) zero-padded FFT periodogram (green line), ii) Welch's method (black line, $N = 32$, $S = 16$), and iii) Marple's method (red line, $p = 15$).

that selecting the appropriate window for a specific application is more of an art than an exact science.

Another solution comes in handy when the signal rides on a relatively high DC level or on a strong sinusoidal signal. In these cases, it is advisable to remove these components from the data before the PSD is estimated. Without taking this precaution, the biasing and strong sidelobes produced could easily obscure weaker components. Whenever physically expected, the DC component of a signal can usually be removed by subtracting the sampled data mean,

$$\bar{x} = \frac{1}{N} \sum_{n=0}^{N-1} x_n,$$

from each data sample to produce a "purely AC" data sequence

$$x_0 - \bar{x}, x_1 - \bar{x}, \dots, x_{N-1} - \bar{x}.$$

ZERO-PADDING THE FFT

An interesting property of the FFT is that simply adding zeros after a windowed-data-samples sequence $x_0, x_1, \dots, x_{N-1}$ to create a longer record $x_0, x_1, \dots, x_{N-1}$

Listing 1—This program estimates the power spectral distribution of a complex data sequence using three different approaches: the zero-padded FFT, the Welch periodogram method, and Marple's autoregressive method. The subroutines listed were adapted from those presented in SL Marple's *Digital Spectral Analysis with Applications*, Englewood Cliffs, NJ: Prentice-Hall, 1987. The subroutines were translated to run under QuickBasic 4.5.

```
DEFDBL A-Z      'Use double precision

' COMPLEX ARRAYS
REM $DYNAMIC:
DIM ar(1),ai(1),dr(1),di(1),rr(1),ri(1),cr(1),ci(1),xr(1),xi(1)
DIM w(1),xfft(1),xfft(1),psd(1),win(1),xsegr(1),xsegi(1)

CLS
INPUT "Please input name of data file : ". filename$
INPUT "Is data complex [y/n] ? ". complex$
INPUT "Sampling period [seconds] ? ". t
INPUT "Periodogram number of samples per segment ? ". nsamp1
INPUT "Periodogram number of samples shift ? ". nshift1
INPUT "Auto regressive model order ? ". ip

' Determine length of data record
n = 0
OPEN filename$ FOR INPUT AS #1
WHILE NOT EOF(1)
  n = n + 1
  IF (complex$ = "y" OR complex$ = "Y") THEN
    INPUT #1, xr(1), xi(1)
  ELSE
    INPUT #1, xr(1)
  END IF
WEND
CLOSE #1
```

(continued)

STOP LOOK LISTEN

Odds are that some time during the day you will stop for a traffic signal, look at a message display or listen to a recorded announcement controlled by a Micromint RTC180. We've shipped thousands of RTC180s to OEMs. Check out why they chose the RTC180 by calling us for a data sheet and price list now.



CALL 1-800-635-3355



MICROMINT, INC.

4 Park Street, Vernon, CT 06066
(203) 871-6170 • Fax (203) 872-2204

in Europe: (44) 0285-658122 • in Canada: (514) 336-9426 • in Australia: (3) 467-7194 • Distributor Inquiries Welcome

Listing 1—continued

```

REDIM xr(n), xi(n) 'Redimension data array
' Read data into array
OPEN filename$ FOR INPUT AS #1
FOR k = 1 TO n
  IF (complex$ = "y" OR complex$ = "Y") THEN
    INPUT #1, xr(k), xi(k)
  ELSE
    INPUT #1, xr(k)
  END IF
NEXT k
CLOSE #1

' Draw display screen on EGA mode 640x350 with 16 colors
SCREEN 9, 0
CLS
LINE (45, 300)-(562, 300), 3, , &H8888
LINE (45, 250)-(562, 250), 3, , &H8888
LINE (45, 200)-(562, 200), 3, , &H8888
LINE (45, 150)-(562, 150), 3, , &H8888
LINE (45, 100)-(562, 100), 3, , &H8888
LINE (45, 50)-(562, 50), 3, , &H8888
LINE (562, 300)-(562, 50), 3, , &H8888
LINE ((562-45)/2+45, 300)-((562-45)/2+45, 50), 5, , &H8888
LOCATE 22, 2: PRINT "-100";
LOCATE 18, 2: PRINT "-80";
LOCATE 15, 2: PRINT "-60";
LOCATE 11, 2: PRINT "-40";
LOCATE 8, 2: PRINT "-20";
LOCATE 4, 2: PRINT "0 dB";
LOCATE 3, 28: PRINT "RELATIVE POWER SPECTRUM";
IF (complex$ = "y" OR complex$ = "Y") THEN
  npsd = 512
  LOCATE 23, 6: PRINT "-0.5";
  LOCATE 23, 39: PRINT "0";
ELSE
  npsd = 1024
  LOCATE 23, 6: PRINT "0";
  LOCATE 23, 37: PRINT "0.25";
END IF
LOCATE 23, 71: PRINT "0.5";
LOCATE 24, 18: PRINT "FRACTION OF SAMPLING FREQ, Fs="; 1/t; " [Hz]";
LOCATE 1, 4: PRINT "PSD ESTIMATORS : ";
COLOR 11: PRINT "Zero-Padded FFT ";
COLOR 12: PRINT "Welch's Method ";
COLOR 10: PRINT "Marple's Method ";

' Compute zero-padded FFT
nshift = 1
nsamp = n 'Set the periodogram for a single segment, &
wintype = 1 'Use rectangular window, in order to
GOSUB periodogram 'Compute zero-padded FFT through periodogram
col = 11: GOSUB plot 'Plot results in light blue

' Estimate the PSD through Welch's averaged periodogram method
nshift = nshift1 'Periodogram num of sample shift between segs
nsamp = nsamp1 'Periodogram num of samples per segment
wintype = 0 'Apply Hamming window
GOSUB periodogram 'Estimate PSD through Welch's method
col = 12: GOSUB plot 'Plot results in light red

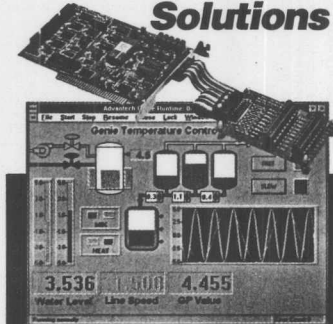
' Estimate the PSD through Marple's method
GOSUB marplepsd 'Estimate PSD through Marple's AR method
col = 10: GOSUB plot 'Plot results in light green
GOTO progend

marplepsd:
' Subroutine to estimate the power spectral distribution of a
' data sequence by Marple's method. This subroutine first solves
' Marple's equations for the estimation of complex autoregressive
' coefficients from complex data. Then, it evaluates the transfer

```

(continued)

ISO 9001 Certified **Data Acquisition & Control Solutions**



Temperature Measurement & Control
Solution Package - \$895

Key Features:

- For J, K, S, T, B, R, E thermocouples
- 8 thermocouple channels, expandable to 32 channels
- 0.1° C resolution with 12-bit A/D
- Up to 250 samples per second
- 16 D/I and 16 D/O channels
- Powerful Windows SCADA software
- Icon based, no programming required

- General Purpose DAS Card\$295
8 A/D, 1 D/A, 16/16 DIO, wiring kit
- Multifunction DAS Card\$395
16 A/D, 2 D/A, 32 DIO, counter
- High Performance DAS Card\$595
100KHz, programmable gain
- 24 Ch Digital I/O Card\$100
- 24 Ch Digital Input w/Interrupt ...\$180
- 32 Ch Digital I/O Card\$170
- 32 Ch Digital Input w/Interrupt ...\$235
- 144 Ch Digital I/O Card\$295
- 8 Relay & 8 Digital Input Card ...\$210
- High Speed Data Streaming Pkg. \$595
100KHz data streaming to disk
- PC Strip Chart Recorder Pkg. ...\$695
16 Channel recording at 15KHz
- Remote DA&C Starter Kit\$495
RS-485 based, up to 4000 feet

FREE!

232-page Solution Guide for
quality minded, budget
conscious Engineers

1-800-800-6889



ADVANTECH.

750 East Arques Ave., Sunnyvale, CA 94086
Tel: (408) 245-6678 FAX: (408) 245-8268

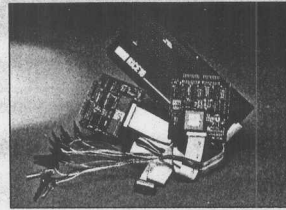
PIC16C5x/16Cxx Real-time Emulators

Introducing RICE16 and RICExx-Juniors, real-time in-circuit emulators for the PIC16C5x and PIC16Cxx family microcontrollers: affordable, feature-filled development systems from **\$599***

* Suggested Retail for U.S. only

RICE16 Features:

- Real-time Emulation to 20MHz for 16C5x and 10MHz for 16Cxx
- PC-Hosted via Parallel Port
- Support all oscillator types
- 8K Program Memory
- 8K by 24-bit real-time Trace Buffer
- Source Level Debugging
- Unlimited Breakpoints
- External Trigger Break with either "AND/OR" with Breakpoints
- Trigger Outputs on any Address Range
- 12 External Logic Probes
- User-Selectable Internal Clock from 40 frequencies or External Clock
- Single Step, Multiple Step, To Cursor, Step over Call, Return to Caller, etc.
- On-line Assembler for patch instruction
- Easy-to-use windowed software
- Support 16C71, 16C84 and 16C64 with Optional Probe Cards
- Comes Complete with TASM16 Macro Assembler, Emulation Software, Power Adapter, Parallel Adapter Cable and User's Guide
- 30-day Money Back Guarantee
- Made in the U.S.A.



Emulators for 16C71/84/64 available now!

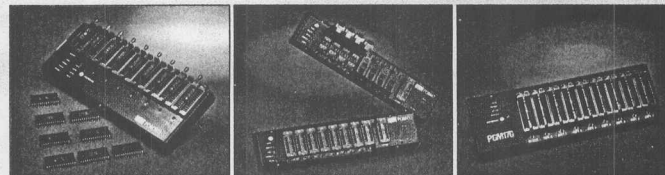
RICE-xx Junior series

RICE-xx "Junior" series emulators support PIC16C5x family, PIC16C71, PIC16C84 or PIC16C64. They offer the same real-time features of RICE16 with the respective probe cards less real-time trace capture. Price starts at \$599.

PIC Gang Programmers

Advanced Transdata Corp. also offers PRODUCTION QUALITY gang programmers for the different PIC microcontrollers.

- Stand-alone COPY mode from a master device
- PC-hosted mode for single unit programming
- High throughput
- Checksum verification on master device
- Code protection
- Verify at 4.5V and 5.5V
- Each program cycle includes blank check, program and verify eight devices
- Prices start at **\$599**



PGM16G: for 16C5x family

PGM47: for 16C71/84

PGM17G: for 17C42

Call (214) 980-2960 today for our new catalog.

For RICE16.ZIP and other product demos, call our BBS at (214) 980-0067.



Advanced Transdata Corporation Tel (214) 980-2960
14330 Midway Road, Suite 128, Dallas, Texas 75244 Fax (214) 980-2937

$x_1, \dots, x_{N-1}, 0, \dots, 0$ before performing the FFT causes the FFT to interpolate transform values between the N original transform values. This process, called *zero padding*, is often mistakenly thought of as a trick to improve the inherent resolution of FFT. Zero padding, however, provides a much smoother PSD and helps annul ambiguities regarding the power and location of peaks that may be scalloped by the nonzero-padded FFT.

CLASSICAL METHODS

As mentioned before, a common mistake is to assume that the solution to Equation (2), the so-called *periodogram*, is a reliable estimate of PSD. Actual proof of this is beyond the scope of this article. But, it has been demonstrated that regardless of how large N is (the number of available data samples), the statistical variance of the estimated periodogram spectrum does not tend to zero. This statistical inconsistency is responsible for the lack of reliability of the periodogram as a spectral estimator.

The solution to this problem is simple, however. If a number of periodograms are computed for different segments of a data record, their average results in a PSD estimate with good statistical consistency. Based on this, Welch [1] proposed a simple method to determine the average of a number of periodograms computed by overlapping segments of the available data record.

Welch's PSD estimate $\hat{P}(f)$ of M data samples is the average of K periodograms $P(f)$ of N points each:

$$\hat{P}(f) = \frac{1}{K} \sum_{i=1}^K P(f)$$

where $P(f)$'s are obtained by applying Equation (2) on appropriately weighted data.

It is obvious that, if the original M -point data record is divided into segments of N points each, with a shift of S samples between adjacent segments, the number of periodograms that can be averaged is:

$$K = \text{Integer} \left(\frac{M-N}{S+1} \right)$$

Listing 1—continued

```

' function of the estimated AR system by using the FFT.
'
' Input Parameters:
'   n - Number of data samples (integer)
'   ip - Order of linear prediction model (integer)
'   xr,xi - Array of complex data samps xr(1),xi(1) - xr(n),xi(n)
'   npsd - Power spectral distribution length
'
' Intermediate Parameters:
'   p - Real linear prediction variance at order ip
'   ar,ai - Array of complex linear prediction coefficients
'
' Output Parameters:
'   psd - Array containing real power spectral distribution,
'         with a maximum power of psdmax
'
REDIM ar(ip), ai(ip), dr(ip + 1), di(ip + 1), rr(ip), ri(ip)
REDIM cr(ip + 1), ci(ip + 1)

r1 = 0
FOR k = 2 TO n - 1
    r1 = r1 + 2 * (xr(k) ^ 2 + xi(k) ^ 2)
NEXT k
r2 = xr(1) ^ 2 + xi(1) ^ 2
r3 = xr(n) ^ 2 + xi(n) ^ 2
r4 = 1 / (r1 + 2 * (r2 + r3))
p = r1 + r2 + r3
delta = 1 - r2 * r4
gamma = 1 - r3 * r4
lambdar = r4 * (xr(1) * xr(n) - xi(1) * xi(n))
lambdai = -r4 * (xr(1) * xi(n) - xr(n) * xi(1))
cr(1) = xr(n) * r4: ci(1) = xi(n) * r4
dr(1) = r4 * xr(1): di(1) = -r4 * xi(1)
m = 0
IF ip = 0 THEN
    p = (.5 * r1 + r2 + r3) / n
    LOCATE 1, 1: BEEP: PRINT "ERROR: Zero AR model order !"
    GOTO progend
END IF

' Main loop of Marple's Modified Covariance algorithm
marpleloop:
m = m + 1
savelr = 0
saveli = 0
FOR k = m + 1 TO n
    savelr = savelr + xr(k) * xr(k - m) + xi(k) * xi(k - m)
    saveli = saveli - xr(k) * xi(k - m) + xi(k) * xr(k - m)
NEXT k
savelr = 2 * savelr: saveli = 2 * saveli
rr(m) = savelr: ri(m) = -saveli
thetar = xr(n)*dr(1)-xi(n)*di(1): thetai = xr(n)*di(1)+dr(1)*xi(n)
psir = xr(n)*cr(1)-xi(n)*ci(1): psii = xr(n)*ci(1)+cr(1)*xi(n)
xir = xr(1)*dr(1)+xi(1)*di(1): xii = xr(1)*di(1)-xi(1)*dr(1)

IF m <> 1 THEN
    FOR k = 1 TO m - 1
        thetar = thetar + xr(n-k) * dr(k+1) - xi(n-k) * di(k+1)
        thetai = thetai + xr(n-k) * di(k+1) + xi(n-k) * dr(k+1)
        psir = psir + xr(n-k) * cr(k+1) - xi(n-k) * ci(k+1)
        psii = psii + xr(n-k) * ci(k+1) + xi(n-k) * cr(k+1)
        xir = xir + xr(k+1) * dr(k+1) + xi(k+1) * di(k+1)
        xii = xii + xr(k+1) * di(k+1) - xi(k+1) * dr(k+1)
        rr(k) = rr(k) - (xr(n+1-m)*xr(n+1-m+k)+xi(n+1-m)*xi(n+1-m+k))
        ri(k) = ri(k) - (-xr(n+1-m)*xi(n+1-m+k)+xi(n+1-m)*xr(n+1-m+k))
        rr(k) = rr(k) - (xr(m)*xr(m-k)+xi(m)*xi(m-k))
        ri(k) = ri(k) - (xr(m)*xi(m-k)-xi(m)*xr(m-k))
        savelr = savelr+rr(k)*ar(m-k)+ri(k)*ai(m-k)
        saveli = saveli+rr(k)*ai(m-k)-ri(k)*ar(m-k)
    NEXT k
END IF

```

(continued)

HIGH-RESOLUTION METHODS

The main limitation of FFT-based methods is restricted spectral resolution. The highest inherent spectral resolution (in Hz) possible with the FFT is approximately equal to the reciprocal of the time interval (in seconds) over which data for the FFT is acquired. This limitation, which is further complicated by leakage and the picket-fence effect, is most noticeable when analyzing short data records.

It is important to note that short data records not only result because of the lack of data (such as when sampling a short transient at a rate barely enough to satisfy Nyquist's criterion), but also from data sampled from a process which slowly varies with time.

For example, by analyzing the vibrations picked up from an oil-well drill, the operator can monitor the buildup of resonance in the long pipe that carries the torque to the drill bit, avoiding costly damages to the equipment [2]. Although a continuous signal from the vibration transducers is available for sampling, the vibrations on the drill assembly change rapidly, resulting in a limited number of data samples which represent each state of the drill bit. It is here where high-resolution estimates would be desirable, even though the data available is limited.

A number of so-called *high-resolution spectral estimators* have been proposed. These alternative methods do not assume, as the FFT does, that the signal outside of the observation window is merely a periodic replica of that observed. Instead, for instance, the parametric estimator relies on the selection of a model, which suitably represents the process generating the signal, to capture the true characteristics of the data outside of the window. By determining the model's parameters, the theoretical PSD, implied by the model, can be calculated and should represent the signal's PSD.

Many signals encountered in real-world applications are well approximated by a rational transfer function model. For example, human speech can be characterized by the resonances

of the vocal tract that generates it. In turn, these resonances are well represented by the poles of a digital filter. Parameters for the filter can be estimated so that the filter could turn a white-noise input into a signal of interest. From the filter's transfer function, we could easily estimate the PSD of the signal.

Various kinds of filter structures exist and are often classified according to the type of transfer function they implement. An all-pole filter is called an *autoregressive* (AR) model, an all-zero filter is a *moving-average* (MA) model, and the general case of a pole-zero filter is called an *autoregressive-moving-average* (ARMA) model. With the last example, the model best suited for speech is then an AR model.

Although high-resolution estimators have been implemented for all these models, AR-based estimators are the most popular because many computationally efficient algorithms are available. A well-behaved set of equations to determine the AR parameters with a computationally efficient algorithm has been introduced by Marple [3].

In the model of Figure 2, the AR filter coefficients $a_1, a_2, \dots, a_p$ are estimated by Marple's algorithm based on the input data samples $x_0, x_1, \dots, x_{N-1}$. The model assumes that a white-noise source drives the filter in which the output is regressed through a chain of delay elements z^{-1} from which p taps feed the AR coefficients. The system's transfer function can then be computed efficiently through the FFT, resulting in an estimate of the signal's PSD.

The performance of Marple's estimator is startling. Figure 3a presents three spectral estimates obtained from a short 64-point complex-test-data set suggested by Marple. Estimates obtained through the zero-padded FFT periodogram, Welch's averaged periodogram, and Marple's method can be compared to the theoretical spectrum of Figure 3b. Only positive-frequency PSD estimates are shown for clarity.

Notice that the closely spaced components cannot be resolved by either of the classical methods, but

Listing 1—continued

```

clr = -save1r / p; cli = -save1i / p
ar(m) = clr: ai(m) = cli
p = p * (1 - clr ^ 2 - cli ^ 2)
IF m <> 1 THEN
  FOR k = 1 TO m / 2
    mk = m - k
    save1r = ar(k): save1i = ai(k)
    auxr = save1r + clr * ar(mk) + cli * ai(mk)
    ai(k) = save1i - clr * ai(mk) + cli * ar(mk)
    ar(k) = auxr
    IF k <> mk THEN
      ar(mk) = ar(mk) + clr * save1r + cli * save1i
      ai(mk) = ai(mk) - clr * save1i + cli * save1r
    END IF
  NEXT k
END IF
IF m = ip THEN
  p = .5 * p / (n - m)
  GOTO arpsd
END IF
' Time update of C,D vectors and GAMMA,DELTA,LAMBDA scalars
r1 = 1 / (delta * gamma - lambdar ^ 2 - lambdai ^ 2)
clr = (thetar * lambdar + thetai * lambdai + psir * delta) * r1
cli = (-thetar * lambdai + thetai * lambdar + psii * delta) * r1
c2r = (psir * lambdar - psii * lambdai + thetar * gamma) * r1
c2i = (psir * lambdai + psii * lambdar + thetai * gamma) * r1
c3r = (xir * lambdar + xii * lambdai + thetar * delta) * r1
c3i = (-xir * lambdai + xii * lambdar + thetai * delta) * r1
c4r = (thetar * lambdar - thetai * lambdai + xir * gamma) * r1
c4i = (thetar * lambdai + thetai * lambdar + xii * gamma) * r1
FOR k = 1 TO (m - 1) / 2 + 1
  mk = m + 1 - k
  save1r = cr(k): save1i = -ci(k)
  save2r = dr(k): save2i = -di(k)
  save3r = cr(mk): save3i = -ci(mk)
  save4r = dr(mk): save4i = -di(mk)
  cr(k) = cr(k) + clr * save3r - cli * save3i + c2r * save4r - c2i * save4i
  ci(k) = ci(k) + clr * save3i + cli * save3r + c2r * save4i + c2i * save4r
  dr(k) = dr(k) + c3r * save3r - c3i * save3i + c4r * save4r - c4i * save4i
  di(k) = di(k) + c3r * save3i + c3i * save3r + c4r * save4i + c4i * save4r
  IF k <> mk THEN
    cr(mk) = cr(mk) + clr * save1r - cli * save1i + c2r * save2r - c2i * save2i
    ci(mk) = ci(mk) + clr * save1i + cli * save1r + c2r * save2i + c2i * save2r
    dr(mk) = dr(mk) + c3r * save1r - c3i * save1i + c4r * save2r - c4i * save2i
    di(mk) = di(mk) + c3r * save1i + c3i * save1r + c4r * save2i + c4i * save2r
  END IF
NEXT k
r2 = psir ^ 2 + psii ^ 2
r3 = thetar ^ 2 + thetai ^ 2
r4 = xir ^ 2 + xii ^ 2
auxr = psir * lambdar - psii * lambdai
auxi = psir * lambdai + psii * lambdar
aux2r = auxr * thetar + auxi * thetai
r5 = gamma - (r2 * delta + r3 * gamma + 2 * aux2r) * r1
auxr = thetar * lambdar - thetai * lambdai
auxi = thetar * lambdai + thetai * lambdar
aux2r = auxr * xir + auxi * xii
r2 = delta - (r3 * delta + r4 * gamma + 2 * aux2r) * r1
gamma = r5
delta = r2
lambdar = lambdar + c3r * psir + c3i * psii + c4r * thetar + c4i * thetai
lambdai = lambdai - c3r * psii + c3i * psir - c4r * thetai + c4i * thetar
IF p <= 0 THEN GOTO arpsderr
IF (delta <= 0 OR delta > 1 OR gamma <= 0 OR gamma > 1) THEN GOTO arpsderr
r1 = 1 / p
r2 = 1 / (delta * gamma - lambdar ^ 2 - lambdai ^ 2)
efr = xr(m + 1): efi = xi(m + 1)
ebr = xr(n - m): ebi = xi(n - m)
FOR k = 1 TO m
  efr = efr + ar(k) * xr(m + 1 - k) - ai(k) * xi(m + 1 - k)
  efi = efi + ar(k) * xi(m + 1 - k) + ai(k) * xr(m + 1 - k)

```

(continued)

Listing 1—continued

```

    ebr = ebr + ar(k) * xr(n - m + k) + ai(k) * xi(n - m + k)
    ebi = ebi + ar(k) * xi(n - m + k) - ai(k) * xr(n - m + k)
NEXT k
c1r = ebr * r1: c1i = ebi * r1
c2r = efr * r1: c2i = -efi * r1
c3r = (ebr * delta + efr * lambdar - efi * lambdai) * r2
c3i = (-ebi * delta + efr * lambdai + efi * lambdar) * r2
auxr = ebr * lambdar - ebi * lambdai
auxi = ebr * lambdai + ebi * lambdar
c4r = (efr * gamma + auxr) * r2: c4i = (efi * gamma - auxi) * r2
FOR k = m TO 1 STEP -1
    save1r = ar(k): save1i = ai(k)
    ar(k) = save1r + c3r * cr(k) - c3i * ci(k) + c4r * dr(k) - c4i * di(k)
    ai(k) = save1i + c3r * ci(k) + c3i * cr(k) + c4r * di(k) + c4i * dr(k)
    cr(k + 1) = cr(k) + c1r * save1r - c1i * save1i
    ci(k + 1) = ci(k) + c1r * save1i + c1i * save1r
    dr(k + 1) = dr(k) + c2r * save1r - c2i * save1i
    di(k + 1) = di(k) + c2r * save1i + c2i * save1r
NEXT k
cr(1) = c1r: ci(1) = c1i
dr(1) = c2r: di(1) = c2i
r3 = ebr ^ 2 + ebi ^ 2
r4 = efr ^ 2 + efi ^ 2
auxr = efr * ebr - efi * ebi: auxi = efr * ebi + efi * ebr
aux2r = auxr * lambdar - auxi * lambdai
p = p - (r3 * delta + r4 * gamma + 2 * aux2r) * r2
delta = delta - r4 * r1
gamma = gamma - r3 * r1
auxr = efr * ebr - efi * ebi: auxi = efr * ebi + efi * ebr
lambdar = lambdar + auxr * r1: lambdai = lambdai - auxi * r1
IF (p>0 AND delta>0 AND delta<=1 AND gamma>0 AND gamma<=1)
    THEN GOTO marpleloop

arpsderr:
LOCATE 1, 6: BEEP
PRINT "ERROR: Numerical ill-conditioning detected for model order>";
    m - 1;
GOTO progend

arpsd: 'Evaluate the AR model's transfer function
nfft = npsd
REDIM xfftr(nfft), xffti(nfft), psd(npsd)
xfftr(1) = 1: xffti(1) = 0
FOR k = 1 TO ip
    xfftr(k + 1) = ar(k): xffti(k + 1) = ai(k)
NEXT k
FOR k = ip + 2 TO npsd
    xfftr(k) = 0: xffti(k) = 0 'Zero-pad up to npsd
NEXT k
GOSUB fft
psdmax = 0
FOR k = 1 TO npsd
    psd(k) = p * t / (xfftr(k) ^ 2 + xffti(k) ^ 2)
    IF psd(k) > psdmax THEN psdmax = psd(k)
NEXT k
RETURN

fft:
' Subroutine to compute the complex FFT of a complex data series.
'
' Input Parameters:
'   nfft. - FFT size
'   t - Sample interval in seconds
'   xfftr, xffti - Array of nfft complex data samples
'               xfftr(1), xffti(1) to xfftr(nfft), xffti(nfft)
'
' Output Parameters:
'   xfftr, xffti - nfft complex transform values replace original
'                 data samples indexed from k=1 to k=nfft,
'                 representing the frequencies (k-1)/nfft*t [Hz]

```

(continued)

they appear clearly separated in the estimate produced by Marple's method. You may also notice that Marple's estimate is "peaky" even for the smooth continuous spectral components at the far right of the PSD.

The reason for this peakiness is that a purely autoregressive filter generates a spectrum based on pure resonances. Only through the use of a moving-average could these resonances be damped to produce a perfectly smooth spectrum in regions where this is necessary. Although this limitation of AR-based estimators would lead to errors in the actual amplitudes of the PSD components, it is very well suited for the high-resolution detection of periodicities in the signal.

A price must be paid for the increase in resolution and, just as you may suspect, the computational burden of these high-resolution methods far exceed that of a simple FFT. In addition, like the selection of an appropriate window for the classical estimators, the rules for selecting an appropriate model, parameter estimation method, and model order are all but cast in stone.

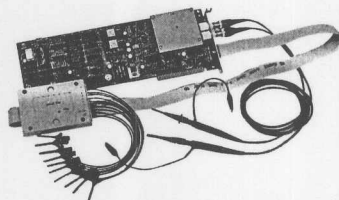
IMPLEMENTING SPECTRAL ANALYSIS ALGORITHMS

Program spectrum.bas presented in Listing 1 demonstrates the implementation of the spectral estimation methods discussed. The program was written in QuickBASIC 4.5, but should run with little trouble under any other BASIC compiler on an IBM PC-compatible with EGA/VGA graphics. However, BASIC does not support complex-number arithmetic, so explicit operations have been used in which variable names with the suffix *r* represent the real portion of that variable, while those with the suffix *i* represent the imaginary portion of the same.

After being defined by the user, the program reads a file containing the *N*-data-point sequence to be analyzed. The data can be either a single column of (plain ASCII) samples or two columns, one containing the real and the other, the imaginary parts of complex data samples. The program

PC-Based Instruments 200 MSa/s DIGITAL OSCILLOSCOPE

**HUGE BUFFER
FAST SAMPLING
SCOPE AND LOGIC ANALYZER
C LIBRARY W/SOURCE AVAILABLE
POWERFUL FRONT PANEL SOFTWARE**



\$1799 - DSO-28204 (4K)
\$2285 - DSO-28264 (64K)

DSO Channels

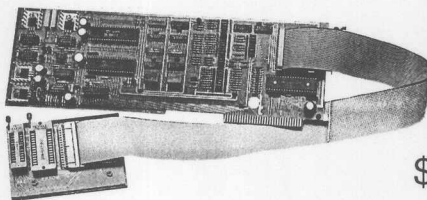
2 Ch. up to 100 MSa/s
or
1 Ch. at 200 MSa/s
4K or 64K Samples/Ch
Cross Trigger with LA
125 MHz Bandwidth

Logic Analyzer Channels

8 Ch. up to 100 MHz
4K or 64K Samples/Ch
Cross Trigger with DSO

Universal Device Programmer

PAL
GAL
EPROM
EEPROM
FLASH
MICRO
PIC
etc..

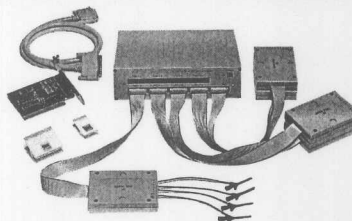


\$475

Free software updates on BBS
Powerful menu driven software

400 MHz Logic Analyzer

- up to 128 Channels
- up to 400 MHz
- up to 16K Samples/Channel
- Variable Threshold Levels
- 8 External Clocks
- 16 Level Triggering
- Pattern Generator Option



\$799 - LA12100 (100 MHz, 24 Ch)
\$1299 - LA32200 (200 MHz, 32 Ch)
\$1899 - LA32400 (400 MHz, 32 Ch)
\$2750 - LA64400 (400 MHz, 64 Ch)

Call (201) 808-8990
Link Instruments
369 Passaic Ave, Suite 100, Fairfield, NJ 07004 fax: 808-8786

will estimate the spectrum of the input data using three methods:

- 1) A single periodogram of the data record is obtained by zero padding the data up to n_{psd} data points ($n_{psd} = 512$ for complex and 1024 for real input data) from which the squared modulus of the FFT is computed. A rectangular window is assumed.
- 2) Welch's method with a Hamming window is applied using the number of samples per periodogram and the shift specified by the user.
- 3) Marple's method is used to estimate the PSD of the data using an AR model with model order given by the user.

Prior to its display in the output screen, PSD is normalized relative to its maximum, and transformed to decibels. For complex input data, both the positive and negative frequency sides of the spectrum are plotted. Otherwise, only the positive frequency spectrum is presented.

Because of screen resolution limitations, the number of computed PSD points for display has been limited to 512. If a larger PSD record is required, however, n_{psd} can be increased to any desired power of 2, and a file can be opened to receive the PSD-estimate results.

A few simple demonstrations can be set up to compare the performance of the methods. First, you may generate a data file for a signal consisting of a single sinusoid at $1/4f_s$ with white noise added to it using the program in Listing 2.

You may vary the signal-to-noise ratio by changing the value of the noise component's coefficient. As well, the frequency of the sinusoidal component may be changed by altering the denominator of the sine argument. Of course, from Nyquist's theorem, a denominator smaller than two produces an aliased signal (you may want to experiment with the effect that this has on the PSD estimate).

In addition, the resolving power of the estimators can be compared by using a signal containing two closely separated sinusoidal components. This

Listing 1—continued

```

REDIM wr(nfft) AS DOUBLE, wi(nfft) AS DOUBLE
' Set up complex exponential table for FFT
nexp = 1
nt = 2 ^ nexp
WHILE nt < nfft
    nexp = nexp + 1
    nt = 2 ^ nexp
WEND
IF nt <> nfft THEN
    LOCATE 1, 4: BEEP: PRINT "Error! nfft is not a power of 2 "
    GOTO progend
END IF
s = 8 * ATN(1) / (nt)
clr = COS(s): cli = -SIN(s)
c2r = 1: c2i = 0
' Compute complex exponential array
FOR k = 1 TO nt
    wr(k) = c2r: wi(k) = c2i
    auxr = c2r * clr - c2i * cli: c2i = c2r * cli + c2i * clr
    c2r = auxr
NEXT k

' Main FFT routine
mm = 1
ll = nfft
FOR k = 1 TO nexp
    nn = ll / 2
    jj = mm + 1
    FOR i = 1 TO nfft STEP ll
        kk = i + nn
        clr = xfftr(i) + xfftr(kk): cli = xffti(i) + xffti(kk)
        xfftr(kk) = xfftr(i) - xfftr(kk): xffti(kk) = xffti(i) - xffti(kk)
        xfftr(i) = clr: xffti(i) = cli
    NEXT i
    IF nn <> 1 THEN
        FOR j = 2 TO nn
            c2r = wr(jj): c2i = wi(jj)
            FOR i = j TO nfft STEP ll
                kk = i + nn
                clr = xfftr(i) + xfftr(kk): cli = xffti(i) + xffti(kk)
                auxr = xfftr(i) - xfftr(kk): auxi = xffti(i) - xffti(kk)
                xfftr(kk) = auxr * c2r - auxi * c2i
                xffti(kk) = auxr * c2i + auxi * c2r
                xfftr(i) = clr: xffti(i) = cli
            NEXT i
            jj = jj + mm
        NEXT j
        ll = nn
        mm = mm * 2
    END IF
NEXT k
nv2 = nfft / 2
nml = nfft - 1
j = 1
FOR i = 1 TO nml
    ii = j + 1
    clr = xfftr(j): cli = xffti(j)
    xfftr(j) = xfftr(ii): xffti(j) = xffti(ii)
    xfftr(ii) = clr: xffti(ii) = cli
    j = j + 1
NEXT j
FOR i = 1 TO nfft
    xfftr(i) = xfftr(i) * t: xffti(i) = xffti(i) * t
NEXT i
RETURN

```

(continued)

NEW Data Acquisition Catalog

Covers expanded low cost line.



FREE!

1994 120 page catalog for PC, VME, and Qbus data acquisition. Plus informative application notes regarding anti-alias filtering, signal conditioning, and more.

NEW Software:
LabVIEW®, LabWindows®, Snap-Master™, and more

NEW Low Cost I/O Boards

NEW Industrial PCs

NEW Isolated Analog and Digital Industrial I/O

New from the inventors of plug-in data acquisition.

Call, fax, or mail for your free copy today.

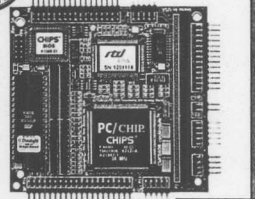
ADAC

American Data Acquisition Corporation
70 Tower Office Park, Woburn, MA 01801
Phone: (800) 648-6589 Fax: (617) 938-6553

#115

Replace Four Conventional PC/104 Modules with One SuperXT™ CMF8680 cpuModule™

Embedded PC/XT Controller with
Intelligent Power Management



PC/104 Compliant
Actual Size: 3.6 x 3.8 x 0.6"

\$449
100 pcs.

- PC/XT compatibility with 286 emulation
- 14 MHz, 16-bit 8086 CPU
- +5V only; 1.6W at 14.3 MHz, 1W at 7.2 MHz
- Intelligent sleep modes, 0.1W in Suspend
- ROM-DOS and RTD enhanced BIOS
- Compatible with MS-DOS & real-time operating systems
- 1M bootable Solid State Disk & free software
- 4K-bit configuration EEPROM (2K for user)
- 2M on-board DRAM
- IDE & floppy interfaces
- CGA CRT/LCD controller
- Two RS-232 ports, one RS-485 port
- Parallel, XT keyboard & speaker ports
- Optional X-Y keypad scanning/PCMCIA interface
- Watchdog timer & real-time clock

Expand This Or Any PC/104 System
with the

CM106 Super VGA Controller utilityModule™

- Mono/color STN & TFT flat panel support
- Simultaneous CRT & LCD operation
- Resolution to 1024 x 768 pixels
- Displays up to 256 colors

\$223
100 pcs.

Speed Product Development with the DS8680 Development System

Your DS8680 includes the CMF8680, CM102 keypad scanning/PCMCIA, CM104 with 1.8" 85MB hard drive, CM106 SVGA controller & DM5406 12-bit, 100 kHz dataModule™ in an enclosure with external power supply, 3.5" floppy, keyboard, keypad, TB50 terminal board, SIGNAL*VIEW™, SIGNAL*MATH™, MS-DOS, SSD software & rtdLinux™ for just

\$2950.

For more information on our PC/104 and
ISA bus products, call today.



Real Time Devices USA

200 Innovation Blvd. • P.O. Box 906

State College, PA 16803 USA

(814) 234-8087 / Fax: (814) 234-5218

RTD Europa • RTD Scandinavia

Real Time Devices is a founder of the PC/104 Consortium.

Listing 1—continued

```

periodogram:
' Subroutine to compute averaged periodogram over nseg segments
' Input Parameters:
'   n      - Number of data samples
'   nshift  - Number of samples shift between segments
'   nsamp   - Number of samples per segment (must be even)
'   t       - Sample interval in seconds
'   xr,xi   - Array of complex samples xr(1),xi(1) to xr(n),xi(n)
'   wintype - Window type - 0 = Hamming, other = rectangular

' Output Parameters:
'   nseg    - Number of segments averaged
'   psd     - Array containing real power spectral distribution,
              with a maximum power of psdmax

REDIM win(nsamp), xsegr(npsd), xsegi(npsd), psd(npsd),
      xfftr(npsd), xffti(npsd)
pi2 = 8 * ATN(1)

' Compute window
FOR k = 1 TO nsamp
  IF wintype = 0 THEN
    ' Hamming window
    win(k) = 0.538 + 0.462 * COS(pi2*(-0.5+(k-1)/(nsamp-1)))
  ELSE
    win(k) = 1 'Rectangular window
  END IF
NEXT k

' Compute Welch's averaged periodogram applying window
nseg = (n - nsamp) / nshift + 1
FOR k = 1 TO nseg
  FOR j = 1 TO nsamp
    index = j + (k - 1) * nshift
    xsegr(j) = xr(index) * win(j)
    xsegi(j) = xi(index) * win(j)
  NEXT j
  FOR j = nsamp + 1 TO npsd
    xsegr(j) = 0: xsegi(j) = 0 'Zero-pad up to npsd
  NEXT j
  nfft = npsd
  FOR j = 1 TO nfft
    xfftr(j) = xsegr(j): xffti(j) = xsegi(j)
  NEXT j
  GOSUB fft
  FOR j = 1 TO npsd
    IF k = 1 THEN
      psd(j) = xfftr(j) ^ 2 + xffti(j) ^ 2
    ELSE
      psd(j) = psd(j) + xfftr(j) ^ 2 + xffti(j) ^ 2
    END IF
  NEXT j
NEXT k
psdmax = 0
FOR k = 1 TO npsd
  psd(k) = psd(k) / (nseg * nsamp)
  IF psd(k) > psdmax THEN psdmax = psd(k)
NEXT k
RETURN

plot:
' Plot results on graph using color col, assuming npsd = 512
' (complex) or npsd = 1024 (real)
FOR k = 1 TO npsd
  psd(k) = 20*LOG(psd(k)/psdmax)/LOG(10) 'Normalize & xform data
  IF psd(k) < -100 THEN psd(k) = -100 'Clip at -100 dB
NEXT k
IF (complex$ = "y" OR complex$ = "Y") THEN
  ' Plot PSD for positive frequencies
  FOR k = 2 TO 256
    LINE(44+k+256,50-2.5*psd(k-1))-(45+k+256,50-2.5*psd(k)),col
  
```

(continued)

Listing 1—continued

```

NEXT k
' Plot PSD for negative frequencies
FOR k = 256 TO 512
    LINE(44+k-256,50-2.5*psd(k-1))-(45+k-256,50-2.5*psd(k)),col
NEXT k
ELSE
' Plot PSD only for positive frequencies
FOR k = 2 TO 512
    LINE(44+k,50-2.5*psd(k-1))-(45+k,50-2.5*psd(k)),col
NEXT k
END IF
RETURN

progend:
' Wait for any key to be pressed to exit program
WHILE INKEY$ = "": WEND

END

```

can be accomplished by adding a second component to the program line which computes x as, for example:

$$x = 2(\text{rnd} - 0.5) + \left(\sin\left(2\pi \frac{i}{4}\right)\right) + \left(\sin\left(2\pi \frac{i}{4.1}\right)\right)$$

A rule of thumb is to keep the AR model order smaller than one-third of the number of available data samples and to allow for at least twice the number of expected spectral components. The code in Listing 1 announces an error whenever mathematical ill-conditioning is encountered due to too-high a model order. However, an unreasonably "peaky" spectrum is often obtained before ill conditioning can be detected.

AN APPLICATION: ARRAY SIGNAL PROCESSING

The greatest interest in high-resolution spectral estimators has been generated in the field known as *array signal processing*. Here, a number of sensors are placed at various locations in space to detect traveling waves.

For example, in seismology, a number of sensors capable of detecting the shock waves of a tremor or earthquake are spread over a certain area. As the shock waves travel under the sensor array, signals from each sensor can be sampled in real time, producing a data record which also contains information regarding the spatial characteristics of the waves (because the sensor locations are known). The processing of resulting spatiotemporal data is meant to estimate the number, vector velocity (speed and direction), and wave shape of the overlapping traveling waves in the presence of interference and noise.

Array signal processing has been successfully applied to many fields besides seismology [4]. For example, in exploration seismogeology, it has been used to image limited regions of the Earth's interior. In passive sonar, it can determine the location and signature of submerged sound sources; in radar, it acquires targets with very high spatial resolution; and in biomedical diagnosis, it tracks weak electrical potentials from the brain, nerves, and

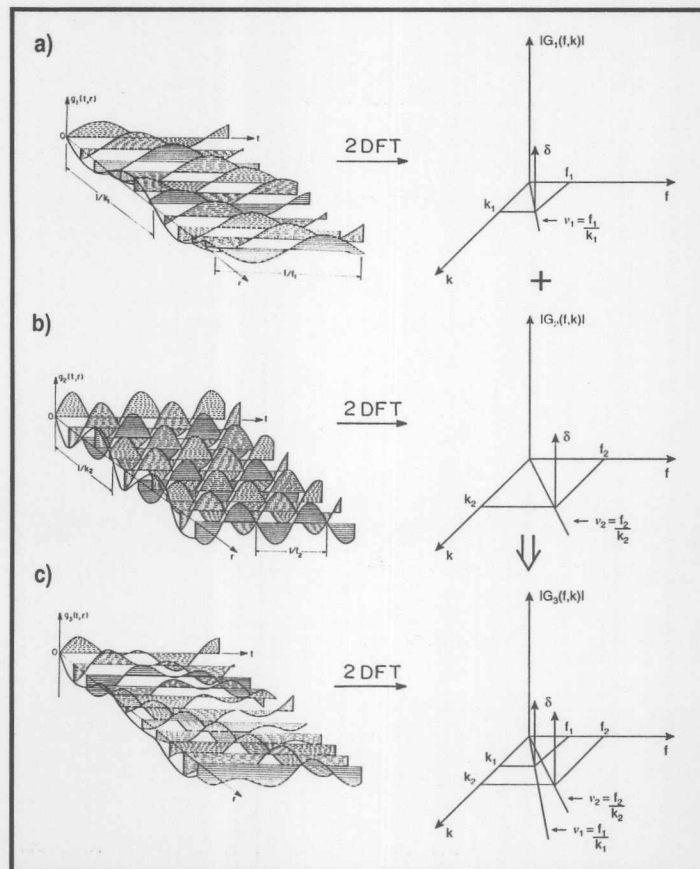


Figure 4—Frequency-wavenumber spectra of planewaves offers (a) spatiotemporal (left) and frequency-wavenumber (right) representations of a sinusoidal wave of frequency f_1 , traveling at propagation velocity v_1 , (b) sinusoidal wave of frequency f_2 , traveling at propagation velocity v_2 , (c) a sum of the above. The two-dimensional PSDs clearly show the component waveform spectra and propagation velocity.

muscles. Other applications include image reconstruction from projections such as radio astronomy and medical tomography.

The most common form of traveling wave is the planewave. In its simplest form, a planewave is a sinusoidal wave that not only propagates through time t , but also through space. In the direction of propagation r , this wave can be represented by:

$$g(t, r) = A \sin(2\pi(ft - \frac{r}{v})) \quad (6)$$

where A is the amplitude of the wave, f is its temporal frequency ($\text{Hz} = 1/\text{s}$), and v is the velocity (in m/s or other suitable velocity units) at which the wave propagates through space.

If one such simple planewave is sampled discretely along time and space, we would obtain a record similar to that presented in the left side of Figure 4a. As you see, at any given time the spatial sampling of the wave also forms a sinusoid with frequency k_1 . The spatial frequency (in $1/\text{m}$) of such a simple planewave is called the *wave number*, and is given by:

$$k = \frac{f}{v} \quad (7)$$

Its physical meaning indicates that at a distance r from the origin, the phase of the wave accumulates by $2\pi kr$ radians. The two-dimensional spectrum of the planewave in our example would be an impulse δ (the spectrum of a sinusoid) located in the frequency-wave number ($f-k$) plane at f_1, k_1 . Through this kind of spectral analysis, we infer the components of the waveform and their velocity because the slope at which the components are found is equal to their propagation velocity or, in this case,

$$v_1 \left[\frac{\text{m}}{\text{s}} \right] = \frac{f_1 \left[\frac{1}{\text{s}} \right]}{k_1 \left[\frac{1}{\text{m}} \right]}$$

Adding a second component (Figure 4b) with a different frequency and propagation velocity to the original component, we obtain a planewave (Figure 4c) that, regardless of its simplicity, can hardly be recognized in the space-time domain.

Listing 2—This program generates a file containing data synthesized from a single sinusoidal signal contaminated by random noise.

```
pi = 3.14159262
OPEN "noise.dat" FOR OUTPUT AS #1
FOR i = 1 TO 256
  x = 2 * (RND - .5) + (SIN(2 * pi * i / 4))
  PRINT #1, x
NEXT i
CLOSE #1
```

However, the two-dimensional frequency-wave number spectrum of the signal clearly resolves the components and their propagation velocities.

The two-dimensional spectrum can be computed with ease knowing that the two-dimensional DFT is computed as a sequence of one-dimensional DFTs of the columns of the data array, followed by a sequence of one-dimensional DFTs of the rows of this new array, or vice versa. As such, the most simple two-dimensional PSD estimator is implemented through the FFT. In practice, however, due to the limited number of spatial samples (because only a few sensors

are normally used), the use of high-resolution estimators is essential.

Considering that enough samples $x_0, x_1, \dots, x_{N-1}$ can usually be obtained from each of the R sensors through time, a hybrid two-dimensional spectral estimator can be implemented by combining the classical and the high-resolution spectral estimation approaches. As shown in Figure 5, using spatiotemporal data $g(t, r)$,

$$g(t, r) = \begin{bmatrix} x_{0,0} & x_{1,0} & \dots & x_{N-1,0} \\ x_{0,1} & x_{1,1} & \dots & x_{N-1,1} \\ \vdots & \vdots & \ddots & \vdots \\ x_{0,R-1} & x_{1,R-1} & \dots & x_{N-1,R-1} \end{bmatrix} \quad (8)$$

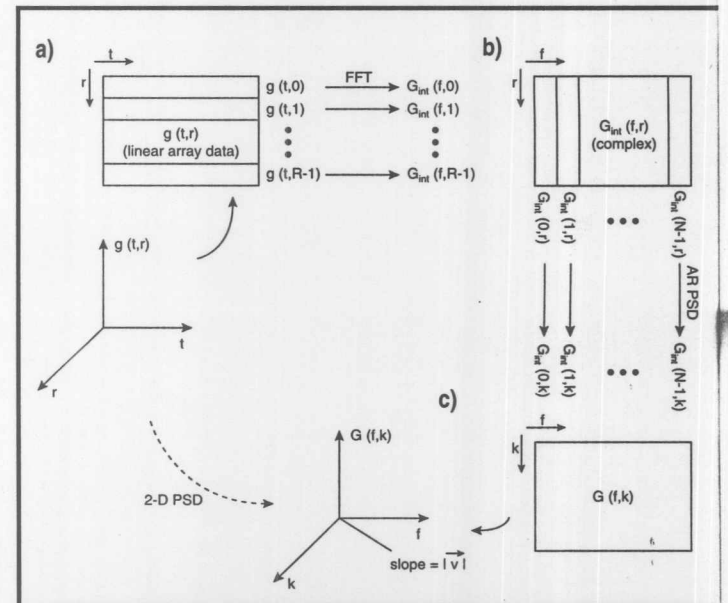


Figure 5—These images illustrate a hybrid two-dimensional spectral estimator. Spatiotemporal data (a) is transformed along time-domain into an intermediate array (b) through the application of a windowed FFT to each and every row of the original data. Applying an AR-PSD estimator to every column of the intermediate array completes the two-dimensional PSD estimation process.

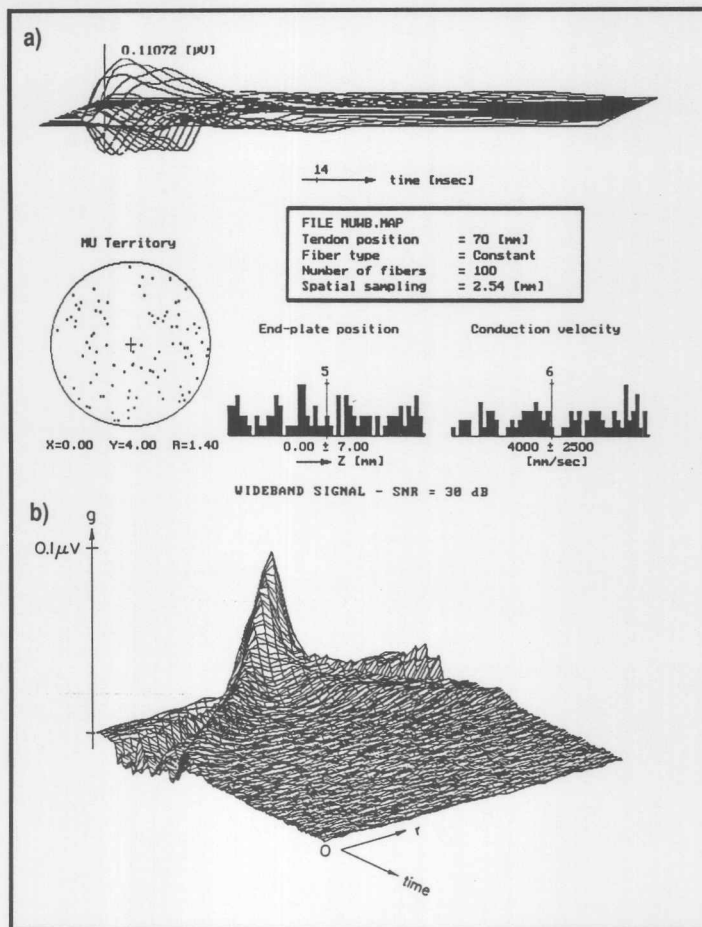


Figure 6—Frequency-wavenumber spectral estimation has been applied to the analysis of the potentials recorded from a muscle twitch. In (a), the complex spatiotemporal waveform has been analyzed to show information about the conduction velocity, origin, and location of the component potentials; (b) shows a magnified view of the spatiotemporal data.

an intermediate transform $G_m(f, r)$ is computed by applying the FFT along each row (time domain) of appropriately weighted data. The two-dimensional spectral estimate $g(f, k)$ is then completed by obtaining the AR-PSD of each column of complex numbers in the intermediate transform.

In the more general case, using an array of sensors spread out over an area with a planewave traveling in any direction under the array, a three-dimensional hybrid spectral estimator determines not only the wave's components and its velocities, but also each component's bearing.

For example, tiny electrical potentials can be picked up from muscle fibers using electrodes attached to the skin. These potentials are caused by pulses (action potentials) that travel down every muscle fiber causing the contraction of muscles. The conduction velocity as well as the origin of these potentials enclose a wealth of information which can be used as an aid in the early diagnosis of nerve and muscle diseases. The large number of convoluted signals and the very small differences between their waveforms makes it impossible to determine this information from

spatiotemporal data (Figure 6b). However, a complete analysis (Figure 6a) is possible through the use of multidimensional spectral estimates.

IN CONCLUSION

Of course, the BASIC program listed here may be too slow to cope with most real-time applications, but implementing both classical and high resolution methods on DSP is a relatively easy task. First of all, modern DSP chips are specifically designed to perform the convolution, vector arithmetic, and FFT operations in a minimal number of clock cycles. In addition, optimized subroutines implementing the most popular high-resolution algorithms are available often in the public domain.

Multidimensional PSD estimation has a very high intrinsic parallelism because spectral estimates are taken independently for every dimension and, as such, can be solved efficiently in parallel. In other words, since tasks in array-signal processing require specific operations to be performed on innumerable data blocks, a parallel system exploits the full power of a number of processors working concomitantly on different portions of the data to solve the larger problem.

High-power computational engines (e.g., Intel's i860) and floating-point DSPs (e.g., Texas Instruments' TMS320C30 and the AT&T DSP32) possess the raw floating-point performance necessary to efficiently implement the relevant algorithms. Unfortunately, however, these chips do not present the flexibility required to implement multiprocessor architectures which can optimally exploit intrinsic parallelism. Moreover, parallel DSP systems using these chips would most likely encounter serious communication bottlenecks imposed by their classical bus-based architectures. In these cases, RISC chips, such as the Transputer family, or DSP chips, such as the TM5320C40, which are designed for parallelism, display the full power of a scalable and very flexible architecture.

I have tried to show you that spectral analysis is a very convenient tool that serves a number of engineer-

ing applications. Moreover, with today's PCs, you have the power to implement modern PSD estimation algorithms with sufficient efficiency for experimenting and even for some real applications. With the enhanced capabilities of DSP chips, PCs with DSP coprocessors and laboratory spectrum analyzers with embedded DSPs become truly powerful and useful instruments.

However, as you understand by now, obtaining good spectral estimates is not only a matter of blindly applying the algorithm and watching the screen. Rather, knowledge about the spectral estimation methods and empirical experience of their use are of foremost importance in obtaining consistent results. □

David Prutchi has a Ph.D. in Biomedical Engineering from Tel-Aviv University. He is an engineering specialist at Intermedics, and his main R&D interest is biomedical signal processing in implantable devices. He may be reached at davidp@mails.imed.com.

REFERENCES

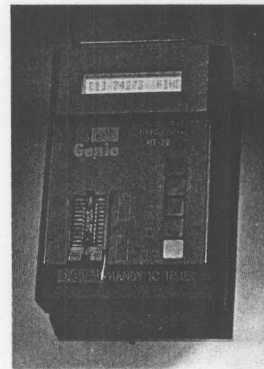
1. Welch, P.D., "The Use of a Fast Fourier Transform for the Estimation of Power Spectra: A Method Based on Time Averaging over Short Modified Periodograms," *IEEE Trans. Audio Electroacoust.*, 15(6), 1967, pp. 70-73.
2. Jangi, S. and Y. Jain, "Embedding Spectral Analysis in Equipment," *IEEE Spectrum*, Feb. 1991, pp. 40-43.
3. Marple, S.L. Jr., *Digital Spectral Analysis with Applications*, Englewood Cliffs, NJ: Prentice-Hall, 1987.
4. Haykin, S. ed., *Array Signal Processing*, Englewood Cliffs, NJ: Prentice-Hall, 1985.

IRS

404 Very Useful
405 Moderately Useful
406 Not Useful

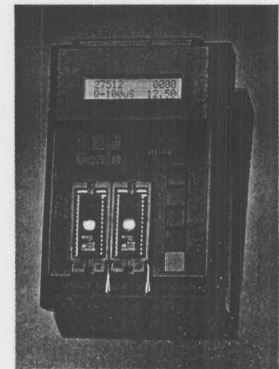
DATA Genie™

Data Genie offers a full line of test & measurement equipment that's innovative, reliable and very affordable. The "Express Series" of stand-alone, non-PC based testers are the ultimate in portability when running from either battery or AC power. Data Genie products will be setting the standards for quality on the bench or in the field for years to come.



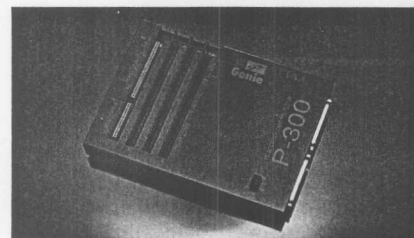
HT-28 Express DIGITAL IC TESTER

The HT-28 is a very convenient way of testing Logic IC's and DRAMs. Tests most TTL 74, CMOS 40/45 and DRAMs 4164-414000, 44164-441000. It can also identify unknown IC numbers on TTL 74 and CMOS 40/45 series with the "Auto-Search" feature.
\$189.95



HT-14 Express EPROM PROGRAMMER

The HT-14 is one-to-one EPROM writer with a super fast programming speed that supports devices from 2732B to 27080, with eight selectable programming algorithms and six programming power (VPP) selections.
\$289.95



P-300

PC INTERFACE CARD PROTECTOR

The Data Genie P-300 is a useful device that allows you to quickly install add-on cards or to test prototype circuits for your PC externally. Without having to turn off your computer to install an add-on card, the P-300 maintains complete protection for your motherboard via the built-in current limit fuses.
\$349.95

MING
Microsystems
Division of MING & P. INC.™
17921 Rowland Street
City of Industry, CA 91748
TEL: (818) 912-7756
FAX: (818) 912-9598

Call for a dealer near you.
1-800-473-6606

Data Genie products are backed by a full 1 year limited factory warranty.

©1994 MING Microsystems. All rights reserved. Data Genie logo is a registered trademark of MING Microsystems.

CCDG01